1992

On Global Transaction Scheduling Criteria in Multidatabase Systems

Aidong Zhang

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Report Number:
92-039

ON GLOBAL TRANSACTION SCHEDULING CRITERIA IN MULTIDATABASE SYSTEMS

Aidong Zhang
Ahmed K. Elmagarmid

CSD-TR-92-039
July 1992

On Global Transaction Scheduling Criteria in Multidatabase Systems *

Aidong Zhang and Ahmed K. Elmagarmid
Department of Computer Science
Purdue University
West Lafayette, IN 47907
{ake,azhang}@cs.purdue.edu

Abstract

This paper presents a theoretical basis for global transaction scheduling to maintain global serializability in multidatabase systems. Three correctness criteria are formulated to utilize the intrinsic characteristics of global transactions at the global level to determine the serialization order of global subtransactions at each local site. In particular, two new types of serializability, chain conflicting serializability and sharing serializability are proposed, and an optimal criterion (called hybrid serializability) combining these two basic criteria is discussed. These criteria offer the advantage of imposing no restrictions on local sites while retaining global serializability. In addition, the optimal aspect of hybrid serializability defines limits on global serializability in multidatabase systems.

1 Introduction

The difficulty of maintaining global serializability in multidatabase systems (MDBSs) with autonomous local database management systems has been evident in the recent literature [AGMS87, BS88, Pu88, DE89, ED90]. This difficulty arises primarily from the constraints posed by the autonomy of local database systems. Various aspects of autonomy, such as design, execution, and control, have been studied in [GMK88, Vei90] and their effects on concurrency control are discussed in [DELO89]. The integration of autonomous local database

*Sponsored by the Indiana Business Modernization and Technology Corporation(BMT), a PYI Award from NSF under grant IRI-8857952, grants from the AT&T Foundation, Bellcore, Tektronix, SERC, Mobil Oil and Bell Northern Research.

systems, each with its own concurrency controller, into a multidatabase via a global concurrency controller inevitably gives rise to a hierarchical structure of global concurrency control. At the lower level, local concurrency controllers, maintain local serializability at local sites, while at the higher level, the global concurrency controller maintains global serializability. These two levels are highly interrelated. Global subtransactions are received by the local concurrency controller and treated as local transactions. The global concurrency controller, on the other hand, must reflect the serialization orders in a manner which is consistent with the local counterparts. In other words, the serialization order of global subtransactions in a local concurrency controller must somehow be reflected or inherited by the global concurrency controller. Thus, the most fundamental issue of global serializability is whether and how the global concurrency controller can determine the serialization order of global subtransactions at each local site without violating local autonomy.

Some approaches to the above issue propose to relax the global serializability theory and simplify global concurrency control. These approaches, such as quasi-serializability [DE89] and two-level serializability [MRKS91], can maintain global consistency in restricted applications. For example, the requirement of no value dependency among sites is allowed in quasi-serializability and restricted Read-Write models are employed in two-level serializability. Other methods impose special restrictions on local database management systems. These approaches, such as rigorous local schedules [BGRS91] or strongly recoverable local schedules [BS92], have achieved some initial success. They may, however, place excessive demands on local database management systems to provide a uniform functionality, such as rigorousness or strong recoverability. The Optimistic Ticket Method (OTM) proposed in [GRS91] is the first to successfully show that the serialization order of global subtransactions in a local site can be determined at the global level without violating local autonomy[1].

In this paper, we provide a theoretical basis for global transaction scheduling to maintain global serializability. In particular, we address the scenario in which the local databases are required only to ensure serializability. Specifically, we attempt to answer the following:

(i) What are the sufficient conditions for the global controller to determine the serialization orders of global subtransactions at local sites without imposing additional restrictions on local database systems; and

(ii) What is the weakest sufficient condition on global transaction scheduling approaches.

We will therefore seek to determine the maximal set of global serializable schedules that can be developed in an MDBS without violating local autonomy. In general, the global concurrency controller has no information about the local serialization orders, and the execution orders of global subtransactions may differ from their serialization orders at local sites. It

[1] In [ED90, MRB+92], an approach which utilizes the information of serialization events or serialization functions contained in local concurrency control protocols is proposed to solve the problem. However, such information may not be generally available.

has been pointed out [DE89, GRS91] that local indirect conflict is the major factor in these discrepancies. Hence, the key to approaching the above two questions is the avoidance of the problem caused by local indirect conflicts. This paper proposes to use novel global scheduling criteria to achieve this goal. Two basic criteria for global transaction scheduling, chain conflicting serializability and sharing serializability, are introduced, and hybrid serializability, an optimal criterion which combines these two basic criteria, is proposed. The optimal aspect of hybrid serializability indicates the maximal class of global transactions that may be scheduled at the global level to maintain global serializability.

The remainder of this paper is organized as follows. Section 2 introduces the system model, defines the relevant terminology, and presents the background of the problem. Sections 3 and 4 discuss, in turn, the two basic criteria of global transaction scheduling, chain conflicting serializability and sharing serializability. In Section 5, hybrid serializability, which combines the features of two basic criteria, is analyzed and its optimality is discussed. In Section 6, a comparison of our work with other related work is given. A final conclusion is provided in Section 7.

2 Preliminaries

In this section, we shall provide a precise definition of the system underconsideration, introduce basic notations and terminology, and discuss the background of the problem.

2.1 The System Model

An MDBS consists of a set of $\{LDBS_i$, for $1 \leq i \leq m\}$, where each $LDBS_i$ is a pre-existing autonomous database management system on a set of data items D_i, superimposed on which is a global database management system (GDBS). Figure 1 depicts the model.

Global transactions are submitted to the GDBS, while local transactions are submitted to LDBSs. Furthermore, as stated in [GPZ86], global serializability cannot be maintained in MDBSs if a global transaction has more than one subtransaction at a given local site. Thus, we assume that each global transaction has at most one subtransaction at each local site.

As a necessary assumption of global serializability, we also presume that the concurrency control mechanisms of LDBSs ensure local serializability. However, no restriction is imposed on these mechanisms.

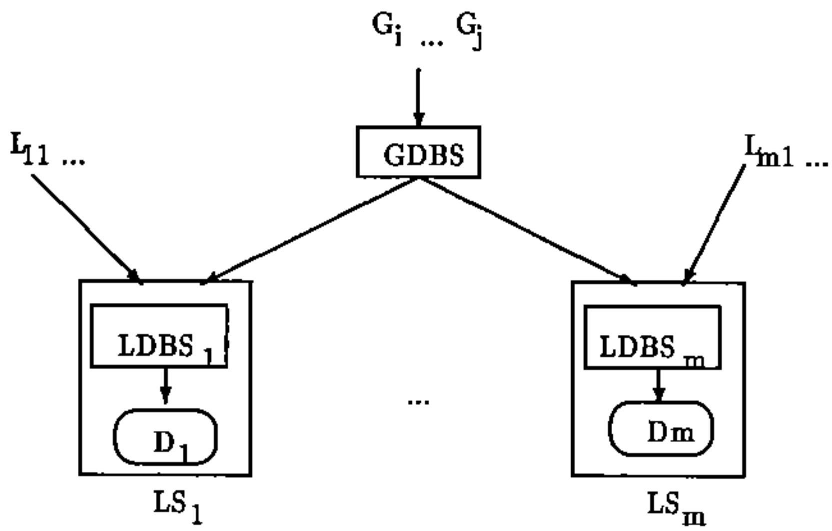


Figure 1: The conceptual architecture for an MDBS

2.2 Notations and Terminology

For the elements of a transaction, we assume the availability of four basic operations: $r(x), w(x), c$, and a, where c and a are *commit* and *abort* termination operations, and $r(x)$ and $w(x)$ are *read* and *write* operations in a local database. Two operations *share* with each other if they access the same data item. Two operations *conflict* with each other if they are sharing operations and at least one of them is a *write* operation.

A local transaction is a partial order of read, write, commit, and abort operations which must specify the order of conflicting operations and contain exactly one termination operation that is the maximum (last) element in the partial order. A more formal definition of a local transaction can be found in [BHG87, Had88]. A set $\mathcal{L}_i = \{L_{i1}, L_{i2}, \cdots, L_{ij_i}\}$ of local transactions contains those local transactions that are submitted directly by local users to $LDBS_i$. A global transaction is a transaction that accesses one or more than one LDBS. The global transaction G_i consists of a set of global subtransactions $\{G_{ij_1}, G_{ij_2}, \cdots, G_{ij_r}\}$, where the subtransaction G_{ij_l} $(1 \leq l \leq r)$ is a local transaction accessing $LDBS_{j_l}$. A set $\mathcal{G} = \{G_1, \cdots, G_n\}$ contains those global transactions that are submitted to the GDBS, and $\mathcal{G}_k$ denotes the set of global subtransactions of $\mathcal{G}$ at local site LS_k. A transaction T_i refers to either a local or global transaction, and OP_{T_i} denotes the set of operations contained in T_i.

Two local transactions T_i and T_j conflict, denoted $T_i \overset{c}{\sim} T_j$, if there exist conflicting operations o_i and o_j such that $o_i \in OP_{T_i}$ and $o_j \in OP_{T_j}$.

A schedule over a set of transactions is a partial order of all and only the operations of these transactions which orders all conflicting operations and respects the order of operations specified by the transactions. A more formal definition of a schedule can also be found in [BHG87, Had88]. A local schedule S_k is a schedule over both local transactions and global subtransactions which are executed at the local site LS_k. A global schedule S is the combination of all local schedules. A global subschedule $S_{\mathcal{G}}$ is S restricted to the set $\mathcal{G}$ of global transactions in S. A lower case s refers to either a local or global schedule.

We say that a schedule s is *serial* if the operations of different transactions in s are not interleaved. We say that the execution of T_1 precedes the execution of T_2 in the schedule s if all operations of T_1 are executed before any operation of T_2 in s. Obviously, a total execution order on transactions in a serial schedule can be determined. We denote $o_1 \prec^s_{eo} o_2$ if operation o_1 is executed before operation o_2 in the schedule s. We denote $T_1 \prec^s_{eo} T_2$ if, for transactions T_1 and T_2 in s and every operation $o_1 \in T_1$ and every operation $o_2 \in T_2$, $o_1 \prec^s_{eo} o_2$.

Let s be a schedule and $C(s)$ be s restricted to the committed transactions in s. We say s is serializable[2] if there exists a serial schedule s' and $C(s)$ is (conflict) equivalent[3] to s'. The execution order of transactions in s' is a serialization order of s. Thus, a global schedule s is serializable if and only if s is serializable in a total order on both committed global and local transactions in s. We denote $T_1 \prec^s_{sr} T_2$ if T_1 precedes T_2 in the serialization order of s.

2.3 Global Serialization Theorem

Since a global schedule is the combination of all local schedules, the global serialization order must inherit local serialization orders. On the other hand, the relative serialization order of the global subtransactions of each global transaction at all local sites needs to be synchronized to maintain global serializability [BS88].

Let O be a total order on transactions. We say that an order O' is consistent with O if O' is a subsequence of O. We assume that a global subtransaction takes the same order symbol as that of the global transaction to which it belongs. The following theorem states that a global schedule S is serializable if and only if there exists a total order O on the global transactions in S, such that in each local schedule of S, the serialization order of its global subtransactions is consistent with O.

Theorem 1 (Global serialization theorem) *If S is a global schedule, then S is serializable if and only if there exists a total order O on global transactions in S such that for each local site $LS_k (1 \leq k \leq m)$, the serialization order of global subtransactions in S_k is consistent with O.*

Theorem 1 has been identified in [MRB+92]; its proof is given in Appendix A.

[2] In this paper, serializability refers to conflict serializability.
[3] See the definition given in [BHG87, Had88].

The above theorem shows that the maintenance of global serializability can be reduced to synchronizing the relative serialization orders of global subtransactions of each global transaction at all local sites. This further implies that the serializability of local schedules, on their own, is not sufficient to maintain global serializability, since global subtransactions in different local databases may have different serialization orders.

Though Theorem 1 provides a necessary and sufficient condition to maintain global serializability, due to the constraints of local autonomy, the GDBS may not be able to generate all global schedules satisfying this condition. Our research has sought to identify alternative correctness conditions to be placed on global subschedules to provide sufficient conditions for the GDBS to maintain global serializability without placing restrictions at local sites.

2.4 Effects of Local Indirect Conflicts

In their early work [GPZ86], Gligor and Popescu-Zeletin considered it sufficient to synchronize the serialization orders of global subtransactions which conflict at local sites. It was generally believed that non-conflict global subtransactions had no effect on global serializability. Later results reported in [BS88, DE89] indicated that, due to local indirect conflicts, the execution order of global subtransactions at a local site may not be consistent with their serialization order, even if they do not conflict. The following example illustrates this situation.

Example 1 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b,c are in D_2. The following global transactions are submitted:*

$G_1 : w_{g_1}(a)r_{g_1}(b) \quad G_2 : r_{g_2}(a)w_{g_2}(c)$

Let $L_{2,1}$ be a local transaction submitted at local site LS_2:

$L_{2,1}$: $w_{L_{2,1}}(b)w_{L_{2,1}}(c)$.

Let S_1 and S_2 be local schedules:

$S_1 : w_{g_1}(a)r_{g_2}(a)$,

$S_2 : w_{L_{2,1}}(b)r_{g_1}(b)w_{g_2}(c)w_{L_{2,1}}(c)$,

and $S = \{S_1, S_2\}$. Though the execution orders of global transactions in both local sites are $G_1 \rightarrow G_2$, the serialization order of S_2 is $G_{22} \rightarrow L_{2,1} \rightarrow G_{12}$. The serialization order of global subtransactions at local site LS_2 is not consistent with their execution order, arising from the indirect conflict of G_{22} with G_{12} (since $w_{g_2}(c)$ conflicts with $r_{L_{2,1}}(c)$ and $w_{L_{2,1}}(b)$ conflicts with $r_{g_1}(b)$).

Thus, even though the execution orders of the global subtransactions in all local sites are consistent, they may differ from their serialization orders in local schedules because of local indirect conflicts. Consequently, global serializability is not maintained. Local indirect conflict is thus the major factor in the difficulty of achieving global serializability in MDBSs. Unfortunately, it is impossible to predict local indirect conflicts at the global level without violating local autonomy, since the GDBS has no knowledge of the submissions of local transactions.

This discussion of local indirect conflicts indicates how the characteristics of local transactions determine the serialization order of global subtransactions at local sites. Conversely, we observe that the characteristics of global transactions can also indirectly effect the serialization order of local schedules at local sites. For instance, in Example 1, if G_2 is changed to $r_{g_2}(a)w_{g_2}(c)w_{g_2}(b)$, then at local site LS_2 after $w_{L_{2,1}}(b)r_{g_1}(b)$ is scheduled, $r_{L_{2,1}}(c)$ would have to be scheduled before $w_{g_2}(c)$ to maintain local serializability. Hence, the correct schedule for S_2 is:

$S_2 : w_{L_{2,1}}(b)r_{g_1}(b)r_{L_{2,1}}(c)w_{g_2}(c)w_{g_2}(b)$,

which implies $G_1 \prec_{sr}^{S_2} G_2$. The conflicting characteristic between global subtransactions G_{12} and G_{22} here imposes an indirect effect on the local scheduling. As another instance, in Example 1, if G_2 is $r_{g_2}(a)r_{g_2}(b)$ and the execution of $r_{g_1}(b)$ at site LS_2 precedes the execution of $r_{g_2}(b)$, then $G_1 \prec_{sr}^{S_2} G_2$ will always be assured in LS_2 (note that $G_2 \prec_{sr}^{S_2} G_1$ may be simultaneously true), even though G_{12} and G_{22} do not conflict. This is due to the fact that there is no local transaction L which can conflict with G_{12} and G_{22} such that $G_2 \prec_{sr}^{S_2} L \prec_{sr}^{S_2} G_1$. We will discuss these properties in detail in the next two sections.

3 Chain Conflicting Serializability

In this section, we investigate a correctness criterion on global subschedules which maintains that the execution order of conflicting operations of global subtransactions is identical to the serialization order of the global subtransactions at each local site. This criterion, termed chain conflicting serializability, provides a sufficient condition for the GDBS to synchronize the relative serialization order of global subtransactions of each global transaction at all local sites without imposing any restrictions other than requiring each $LDBS$ to ensure local serializability.

3.1 The principle

Definitions of chain conflicting transactions and chain conflicting serializable schedules will first be provided. We will then show that, if global subschedules are chain conflicting serializable, global serializability is assured. No restriction except local serializability is required at local sites.

Definition 1 (Chain conflicting transactions) *A set T of local transactions is chain conflicting if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $T_{i_1} \overset{c}{\sim} T_{i_2} \overset{c}{\sim} \cdots \overset{c}{\sim} T_{i_n}$. A set $\mathcal{G}$ of global transactions is chain conflicting if there is a total order O on $\mathcal{G}$ such that for all $k, 1 \leq k \leq m$, $\mathcal{G}_k$ is chain conflicting in an order consistent with O.*

Example 2 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b,c are in D_2. Three global transactions are given as follows:*

$G_1 : r_{g_1}(a)w_{g_1}(b)r_{g_1}(c) \qquad G_2 : w_{g_2}(a) \qquad G_3 : r_{g_3}(a)r_{g_3}(b)$

where $\{G_1, G_2, G_3\}$ is chain conflicting in the order $G_1 \rightarrow G_2 \rightarrow G_3$. No other alternative chain conflicting orders exist. Note that G_2 does not have a global subtransaction at local site LS_2.

Note that $T_1 \overset{c}{\sim} T_2 \overset{c}{\sim} T_3$ may not imply $T_1 \overset{c}{\sim} T_3$.

Definition 2 (Chain conflicting serializable schedules) *A schedule s is chain conflicting serializable if the set T of transactions in s is chain conflicting in a total order O on T and s is serializable in O.*

We will now illustrate the application of chain conflicting serializability in the MDBS environment. We give the following main theorem first.

Theorem 2 *Let S be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S. If $S_{\mathcal{G}}$ is chain conflicting serializable, then the local serializability of S_k (for k=1,...,m) implies the global serializability of S.*

The proof of this theorem relies on Lemma 1, which shows that the outcome of a concurrent execution of transactions depends only on the relative ordering of conflicting operations [BHG87].

Lemma 1 *If o_1 and o_2 are conflicting operations of transactions T_1 and T_2 (respectively) in a serializable schedule s, then $o_1 \prec^s_{eo} o_2$ if and only if $T_1 \prec^s_{sr} T_2$.*

Proof: (*if*) We need to show that $T_1 \prec^s_{sr} T_2$ implies $o_1 \prec^s_{eo} o_2$. Suppose $o_1 \not\prec^s_{eo} o_2$. Then $o_2 \prec^s_{eo} o_1$, and in any serial schedule s' which is conflict equivalent to S, $o_2 \prec^{s'}_{eo} o_1$. Hence, $T_1 \not\prec^s_{sr} T_2$.

(*only if*) Conversely, we need to show that $o_1 \prec^s_{eo} o_2$ implies $T_1 \prec^s_{sr} T_2$. Similarly to above, suppose $T_1 \not\prec^s_{sr} T_2$. Then $T_2 \prec^s_{sr} T_1$. Since o_1 conflicts o_2, in any serial schedule s' which is conflict equivalent to s, $T_2 \prec^{s'}_{eo} T_1$, which implies $o_2 \prec^{s'}_{eo} o_1$. Hence, $o_2 \prec^s_{eo} o_1$, which implies $o_1 \not\prec^s_{eo} o_2$. □

We now apply Lemma 1 to the MDBS environment. Assume a global subschedule $S_{\mathcal{G}}$ of global schedule S is serializable in a total order O on $\mathcal{G}$, and $G_i \in \mathcal{G}$ precedes $G_j \in \mathcal{G}$ in O. If, for integer k $(1 \leq k \leq m)$, $G_{ik} \overset{c}{\sim} G_{jk}$ and o_{ik}, o_{jk} are conflicting operations of G_{ik} and G_{jk}, respectively, then by the *"if"* part of Lemma 1, $o_{ik} \prec_{eo}^{S_{\mathcal{G}}} o_{jk}$. Consequently, at local site LS_k, $o_{ik} \prec_{eo}^{S_k} o_{jk}$. If S_k is serializable, then by the *"only if"* part of Lemma 1, $G_{ik} \prec_{sr}^{S_k} G_{jk}$. We have shown that the characteristics of global subschedules can indirectly affect the serialization orders of global subtransactions in local schedules.

We now present the proof of Theorem 2.

Proof: Suppose $S_{\mathcal{G}}$ is chain conflicting serializable in a total order $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$ on $\mathcal{G}$. Without loss of generality, we assume that, at local site LS_k $(1 \leq k \leq m)$, $G_{i_1k}, G_{i_2k}, \cdots, G_{i_nk}$ exist. We need to prove that, if S_k is serializable, then $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$. The proof proceeds by induction on integer n:

n=1: Straightforward.

Suppose for $n = j(\geq 1)$, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_jk}$ holds.

$n = j+1$, since G_{i_j} precedes $G_{i_{j+1}}$ in O, $G_{i_jk} \overset{c}{\sim} G_{i_{j+1}k}$. If o_{i_jk} and $o_{i_{j+1}k}$ are conflicting operations of G_{i_jk} and $G_{i_{j+1}k}$, respectively, then by the *"if"* part of Lemma 1, $o_{i_jk} \prec_{eo}^{S_{\mathcal{G}}} o_{i_{j+1}k}$, which is equivalent to $o_{i_jk} \prec_{eo}^{S_k} o_{i_{j+1}k}$. Then by the *"only if"* part of Lemma 1, $G_{i_jk} \prec_{sr}^{S_k} G_{j+1k}$. Consequently, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$.

Hence, the serialization order of global subtransactions in S_k $(1 \leq k \leq m)$ is consistent with O. Consequently, by Theorem 1, S is serializable. □

Following Theorem 2, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is chain conflicting. In addition, only conflicting operations need to be ordered. A traditional graph-theoretic characterization of chain conflicting serializability for global transaction execution ordering is discussed below.

Let us first introduce the global transaction execution graph.

Definition 3 *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S, and $\mathcal{G}$ is chain conflicting in a total order O on $\mathcal{G}$. The global transaction execution graph of $S_{\mathcal{G}}$ in O, denoted $GEG_c(S_{\mathcal{G}}^O)$, is a directed graph whose nodes are the global transactions in $\mathcal{G}$ and whose edges are all the relations $(G_i, G_j)(i \neq j)$ such that $G_i \rightarrow G_j$ if and only if : (1) G_i precedes G_j in O; or (2) at LS_k $(1 \leq k \leq m)$, there exist conflicting operations $o_{ik} \in OP_{G_{ik}}$, $o_{jk} \in OP_{G_{jk}}$ and $o_{ik} \prec_{eo}^{S_k} o_{jk}$.*

Theorem 3 (Global execution theorem) *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S. If $\mathcal{G}$ is chain conflicting in a total order O on $\mathcal{G}$, then $S_{\mathcal{G}}$ is chain conflicting serializable in O if and only if $GEG_c(S_{\mathcal{G}}^O)$ is acyclic.*

Proof : Let $S = \{S_1, S_2, ..., S_m\}$ be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S and $\mathcal{G}$ is chain conflicting in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$.

(*if*) Since $GEG_c(S_\mathcal{G}^O)$ is acyclic, it can be topologically sorted. Obviously, by the definition of $GEG_c(S_\mathcal{G}^O)$, $G_{i_1}, G_{i_2}, ..., G_{i_n}$ must be the topological sort of $GEG_c(S_\mathcal{G}^O)$. Let $S'_\mathcal{G}$ be the serial schedule $G_{i_1}, G_{i_2}, ..., G_{i_n}$. We claim that $S_\mathcal{G}$ is conflict equivalent to $S'_\mathcal{G}$. To illustrate this, let $op_i \in OP_{G_i}$ and $op_j \in OP_{G_j}$, where G_i, G_j are committed global transactions in S. Suppose op_i and op_j conflict and $op_i \prec_{eo}^{S_\mathcal{G}} op_j$. By the definition of $GEG_c(S_\mathcal{G}^O)$, $G_i \rightarrow G_j$ is an edge in $GEG_c(S_\mathcal{G}^O)$. Thus, in $S'_\mathcal{G}$, all operations of G_i appear before any operation of G_j, and in particular, $op_i \prec_{eo}^{S'_\mathcal{G}} op_j$. Similarly to the proof of the serialization theorem in [BHG87], $S_\mathcal{G}$ is conflict equivalent to $S'_\mathcal{G}$. Hence, $S_\mathcal{G}$ is chain conflicting serializable in O.

(*only if*) Let $S_\mathcal{G}$ be chain conflicting serializable in O. Let $S'_\mathcal{G}$ be a serial schedule $G_{i_1} G_{i_2} ... G_{i_n}$, which is conflict equivalent to $S_\mathcal{G}$. Consider an edge $G_i \rightarrow G_j$ in $GEG_c(S_\mathcal{G}^O)$. We have either G_i precedes G_j in O or there are two conflicting operations o_i,o_j of G_i,G_j(respectively), such that $o_i \prec_{eo}^{S_\mathcal{G}} o_j$. Consequently, it follows that G_i appears before G_j in $S'_\mathcal{G}$ since $S'_\mathcal{G}$ is serial in O and conflict equivalent to $S_\mathcal{G}$. Now suppose there is a cycle in $GEG_c(S_\mathcal{G}^O)$, and without lose of generality let that cycle be $G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_r \rightarrow G_1$ $(r > 1)$. These edges imply that in $S'_\mathcal{G}$, G_1 appears before G_2 which appears before G_3 which appears ... before G_r which appears before G_1. Thus, the existence of the cycle implies that each of $G_1, G_2, \cdots, G_r$ appears before itself in the serial schedule $S'_\mathcal{G}$, that contradicts our assumption. Hence, $GEG_c(S_\mathcal{G}^O)$ is acyclic. □

Example 3 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b,c are in D_2. The following global transactions are submitted:*

$G_1 : w_{g_1}(a) r_{g_1}(b) \quad G_2 : r_{g_2}(a) w_{g_2}(c) w_{g_2}(b) \quad G_3 : w_{g_3}(a) r_{g_3}(c)$

which is chain conflicting in the order $G_1 \rightarrow G_2 \rightarrow G_3$. Let $L_{2,1}$ be a local transaction submitted at local site LS_2:

$L_{2,1}$: $w_{L_{2,1}}(b) w_{L_{2,1}}(c)$

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : w_{g_1}(a) r_{g_2}(a) w_{g_3}(a)$

$S_2 : w_{L_{2,1}}(b) r_{g_1}(b) w_{L_{2,1}}(c) w_{g_2}(c) r_{g_3}(c) w_{g_2}(b)$.

Obviously, $S_\mathcal{G}$ is serializable in the order $G_1 \rightarrow G_2 \rightarrow G_3$, and S is serializable. Note that, as long as the execution orders of conflicting operations of global subtransactions are controlled identically at both local sites, such as:

$$w_{g_1}(a) \prec_{eo}^{S_1} r_{g_2}(a) \prec_{eo}^{S_1} w_{g_3}(a)$$

$$r_{g_1}(b) \prec_{eo}^{S_2} w_{g_2}(b), \quad w_{g_2}(c) \prec_{eo}^{S_2} r_{g_3}(c)$$

then global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems.

In $GEG(S^{O}_{\{G_1,G_2,G_3\}})$, *we have:*

$$\begin{tikzcd} G_1 \arrow[r] \arrow[rr, bend right] & G_2 \arrow[r] & G_3 \end{tikzcd}$$

Note that $G_{12} \overset{c}{\not\to} G_{32}$. *In the following schedule* S'*:*

$S'_1 : w_{g_1}(a)w_{g_3}(a)r_{g_2}(a)$,

$S'_2 : w_{L_{2,1}}(b)r_{g_1}(b)r_{g_3}(c)w_{L_{2,1}}(c)w_{g_2}(c)w_{g_2}(b)$,

S'_G *is serializable (not chain conflicting serializable) in the order* $G_1 \to G_3 \to G_2$, *but* S' *is not serializable.* □

3.2 Forcing Chain Conflicts in Global Transactions

One advantage of chain conflicting serializability is that it can be easily generalized to all global transactions by forcing chain conflicts in global transactions. For example, an elegant method, termed the ticket method, is proposed in [GRS91]. The ticket method introduces a data item called *ticket* in each local site and requires each global subtransaction to access the ticket at its site. Consequently, conflicts are created among all global subtransactions in the same site. The ticket method thus generates an instance which satisfies a strong condition of the chain conflicting property; tickets cause the set of all global transactions to be chain conflicting in any order. A minor problem with the ticket method is that a local site may not be willing to allow the creation of a ticket in its database.

An alternative method, which we will term the extra operation method, may be suggested to circumvent this difficulty. Let G_{ik} and G_{jk} be global subtransactions in local site LS_k which do not conflict. Chain conflicts can then be simulated. Suppose G_{ik} is executed before G_{jk}. If G_{ik} and G_{jk} have no conflict and the last operation of G_{ik} is on data item x, then append operations $r(x)$ and $w(x)$ to G_{jk} (denoted G'_{jk}). Now G_{ik} and G'_{jk} conflict with each other, and the effect on D_k made by G'_{jk} remains the same as that made by G_{jk}. The advantage of the extra operation method is that it requires nothing from local sites. On the other hand, while extra *read* operations would seem reasonable to local sites, these extra *write* operations may appear unnecessary to local sites and would be ignored by the optimization of LDBSs. The GDBS must ensure that these extra updating operations will be executed at local sites. In Section 5, we will show that the need to insert update operations can be avoided.

4 Sharing Serializability

In this section, we investigate another correctness criterion on global subschedules which maintains that the execution order of the sharing operations of global subtransactions is identical to their serialization order in each local site. This criterion, termed sharing serializability, provides another sufficient condition for the GDBS to synchronize the relative serialization order of global subtransactions of each global transaction at all local sites.

4.1 The principle

The definitions of fully sharing transactions and sharing serializable schedules will first be provided. We will then show that, if global subschedules are sharing serializable, global serializability is assured. No restriction except local serializability is required at local sites.

Let D_T denote the set of data items that transaction T accesses.

Definition 4 (Fully sharing transactions) *A set T of local transactions is fully sharing if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $D_{T_{i_1}} \subseteq D_{T_{i_2}} \subseteq \cdots \subseteq D_{T_{i_n}}$. A set $\mathcal{G}$ of global transactions is fully sharing if there is a total order O on $\mathcal{G}$ such that for all $k, 1 \leq k \leq m$, $\mathcal{G}_k$ is fully sharing in an order consistent with O.*

The fully sharing relation of transactions is defined with respect to the data accessed by the transactions other than the types of operations. A set of transactions may be chain conflicting but not fully sharing or it may be fully sharing but not chain conflicting. In Example 2, $\{G_1, G_2, G_3\}$ is fully sharing in the order $G_2 \rightarrow G_3 \rightarrow G_1$. There is no other alternative fully sharing relation.

The execution order of sharing operations of transactions can also determine the serialization order of the transactions, as expressed in the following lemma:

Lemma 2 *Assume that T_1 and T_2 are transactions in a serializable schedule s such that $D_{T_1} \subseteq D_{T_2}$. If, for all sharing operations $o_1 \in OP_{T_1}, o_2 \in OP_{T_2}$, $o_1 \prec_{eo}^{s} o_2$, then $T_1 \prec_{sr}^{s} T_2$.*

Proof: (1) If T_1 and T_2 conflict, then since conflicting operations must access common data, there exist conflicting operations $o_1 \in OP_{T_1}, o_2 \in OP_{T_2}$, $o_1 \prec_{eo}^{s} o_2$. Hence, $T_1 \prec_{sr}^{s} T_2$ follows from Lemma 1.

(2) If T_1 and T_2 do not conflict, then we need to prove that there is no transaction T' which conflicts with T_1, and consequently also conflicts with T_2 (since $D_{T_1} \subseteq D_{T_2}$) such that $T_2 \prec_{sr}^{s} T' \prec_{sr}^{s} T_1$.

The proof proceeds by contradiction. Suppose we do have a transaction T' which conflicts with T_1 and T_2 such that $T_2 \prec_{sr}^{s} T' \prec_{sr}^{s} T_1$. Since $D_{T_1} \subseteq D_{T_2}$, an operation of T' which conflicts with T_1 must also conflict with T_2. Without loss of generality, let o_1, o', and o_2 be conflicting operations of T_1, T', and T_2, respectively. By Lemma 1, we have $o_2 \prec_{eo}^{s} o' \prec_{eo}^{s} o_1$, contradicting the assumption $o_1 \prec_{eo}^{s} o_2$. □

Definition 5 (Sharing equivalence) *Two global subschedules $S_{\mathcal{G}}$ and $S'_{\mathcal{G}}$ of global schedule S and S' are said to be sharing equivalent, denoted $S_{\mathcal{G}} \equiv_s S'_{\mathcal{G}}$, if they have the same operations of $\mathcal{G}$ where $\mathcal{G}$ is fully sharing in a total order O on $\mathcal{G}$ and if G_i precedes G_j in O, then for each integer k $(1 \leq k \leq m)$ and all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_{\mathcal{G}}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_{\mathcal{G}}} o_{jk}$.*

Definition 6 (Sharing serializability) *A global subschedule is sharing serializable if and only if it is sharing equivalent to a serially global subschedule.*

Note that sharing serializability is stronger than serializability; in other words, sharing serializability implies serializability.

In Example 2, a global subschedule $S_{\mathcal{G}} = w_{g_2}(a)r_{g_3}(a)r_{g_1}(a)r_{g_3}(b)r_{g_1}(b)r_{g_1}(c)$ is sharing serializable in the order $G_2 \rightarrow G_3 \rightarrow G_1$.

We will now illustrate the application of sharing serializability in the MDBS environment, addressing first the the application of Lemma 2.

Assume a global subschedule $S_{\mathcal{G}}$ is sharing serializable in a total order O on $\mathcal{G}$, and $G_i \in \mathcal{G}$ precedes $G_j \in \mathcal{G}$ in O. If, for integer k $(1 \leq k \leq m)$, for all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_{\mathcal{G}}} o_{jk}$, then at local site LS_k, $o_{ik} \prec_{eo}^{S_k} o_{jk}$. If S_k is serializable, by Lemma 2, $G_{ik} \prec_{sr}^{S_{L_k}} G_{jk}$. We have shown that the characteristics of global subschedules can indirectly affect the serialization order of global subtransactions in local schedules.

Our major theorem is the following:

Theorem 4 *Let S be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S. If $S_{\mathcal{G}}$ is sharing serializable, then the local serializability of S_k (for k=1,...,m) implies the global serializability of S*[4]*.*

Proof: Suppose $S_{\mathcal{G}}$ is sharing serializable in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$ on $\mathcal{G}$. Without loss of generality, we assume that, at local site LS_k $(1 \leq k \leq m)$, $G_{i_1k}, G_{i_2k}, \cdots, G_{i_nk}$ exist. We need to prove that, if S_k is serializable, then $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$. The proof proceeds by induction on integer n:

n=1: Straightforward.

Suppose for $n = j(\geq 1)$, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_jk}$ holds.

[4] A similar theory can be propounded using the relationship $D_{T_{i_1}} \supseteq D_{T_{i_2}} \supseteq \cdots \supseteq D_{T_{i_n}}$.

$n = j+1$, since G_{i_j} precedes $G_{i_{j+1}}$ in O, then for all sharing operations $o_{i_jk} \in OP_{G_{i_jk}}, o_{i_{j+1}k} \in OP_{G_{i_{j+1}k}}$, $o_{i_jk} \prec_{eo}^{S_{\mathcal{G}}} o_{i_{j+1}k}$, which is equivalent to $o_{i_jk} \prec_{eo}^{S_k} o_{i_{j+1}k}$. By Lemma 2, $G_{i_jk} \prec_{sr}^{S_k} G_{j+1k}$. Consequently, $G_{i_1k} \prec_{sr}^{S_k} G_{i_2k} \prec_{sr}^{S_k} \cdots \prec_{sr}^{S_k} G_{i_nk}$.

Hence, the serialization order of global subtransactions in S_k $(1 \leq k \leq m)$ is consistent with O. Consequently, by Theorem 1, S is serializable. □

Following Theorem 4, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is fully sharing. In addition, only sharing operations need be ordered. This criterion shows that the serialization order of global subtransactions at a local site can be determined at the global level without requiring that the global subtransactions be conflicting. Note that both classes of global subschedules that satisfy chain conflicting serialization or sharing serializability are not disjoint.

A traditional graph-theoretic characterization of sharing serializability for global transaction execution ordering is discussed below.

Let us first introduce the global transaction execution graph.

Definition 7 *Let $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ be committed global transactions in the global schedule S, and $\mathcal{G}$ is sharing serializable in a total order O on $\mathcal{G}$. The global transaction execution graph of $S_{\mathcal{G}}$ in O, denoted $GEG_s(S_{\mathcal{G}}^O)$, is a directed graph whose nodes are the global transactions in S and whose edges are all the relations $(G_i, G_j)(i \neq j)$ such that $G_i \rightarrow G_j$ if and only if : (1) G_i precedes G_j in O; or (2) at LS_k $(1 \leq k \leq m)$, there exist sharing operations $o_{ik} \in OP_{G_{ik}}$, $o_{jk} \in OP_{G_{jk}}$ and $o_{ik} \prec_{eo}^{S_k} o_{jk}$.*

Theorem 5 (Global execution theorem) *Let $\mathcal{G}$ be the set of committed global transactions in the global schedule S. If $\mathcal{G}$ is fully sharing in a total order O on $\mathcal{G}$, then $S_{\mathcal{G}}$ is sharing serializable in O if and only if $GEG_s(S_{\mathcal{G}}^O)$ is acyclic.*

Proof: Let $S = \{S_1, S_2, ..., S_m\}$ be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S and $\mathcal{G}$ is fully sharing in a total order O of $G_{i_1}, G_{i_2}, \cdots, G_{i_n}$.

(if) Since $GEG_s(S_{\mathcal{G}}^O)$ is acyclic, it can be topologically sorted. Obviously, by the definition of $GEG_s(S_{\mathcal{G}}^O)$, $G_{i_1}, G_{i_2}, ..., G_{i_n}$ must be the topological sort of $GEG_s(S_{\mathcal{G}}^O)$. Let $S'_{\mathcal{G}}$ be the serial schedule $G_{i_1}, G_{i_2}, ..., G_{i_n}$. We claim that $S_{\mathcal{G}}$ is conflict equivalent to $S'_{\mathcal{G}}$. To illustrate this, let $op_i \in OP_{G_i}$ and $op_j \in OP_{G_j}$, where G_i, G_j are committed global transactions in S. Suppose op_i and op_j conflict (also share) with each other and $op_i \prec_{eo}^{S_{\mathcal{G}}} op_j$. By the definition of $GEG_s(S_{\mathcal{G}}^O)$, $G_i \rightarrow G_j$ is an edge in $GEG_s(S_{\mathcal{G}}^O)$. Thus, in $S'_{\mathcal{G}}$, all operations of G_i appear before any operation of G_j, and in particular, $op_i \prec_{eo}^{S'_{\mathcal{G}}} op_j$. Similarly to [BHG87], $S_{\mathcal{G}}$ is conflict equivalent to $S'_{\mathcal{G}}$. Hence, $S_{\mathcal{G}}$ is sharing serializable in O.

(only if) Let $S_{\mathcal{G}}$ be sharing serializable in O. Suppose there is a cycle in $GEG_s(S_{\mathcal{G}}^O)$, and without lose of generality let that cycle be $G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_r \rightarrow G_1$ $(r > 1)$. These edges

imply that there exist two global transactions $G_i, G_j (1 \leq i, j \leq r)$ such that G_i precedes G_j in O, but $o_j \prec_{eo}^{S_g} o_i$ for $o_i \in G_i$ and $o_j \in G_j$. That contradicts our assumption that S_g is sharing serializable in O. Hence, $GEG_s(S_g^O)$ is acyclic. $\square$

Example 4 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b,c are in D_2. The following global transactions are submitted:*

$G_1 : w_{g_1}(a) r_{g_1}(b) \quad G_2 : r_{g_2}(a) w_{g_2}(c) r_{g_2}(b),$

which is fully sharing in the order $G_1 \rightarrow G_2$. Let $L_{2,1}$ be a local transaction submitted at local site LS_2:

$L_{2,1}$: $w_{L_{2,1}}(b) r_{L_{2,1}}(c)$

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : w_{g_1}(a) r_{g_2}(a),$

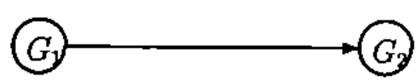
$S_2 : w_{L_{2,1}}(b) r_{g_1}(b) r_{L_{2,1}}(c) w_{g_2}(c) r_{g_2}(b).$

Obviously, S_g is serializable in the order $G_1 \rightarrow G_2$, and S is serializable. Note that G_{12} and G_{22} do not conflict. However, as long as the execution orders of sharing operations of global subtransactions are controlled in the order:

$$w_{g_1}(a) \prec_{eo}^{S_1} r_{g_2}(a)$$

$$r_{g_1}(b) \prec_{eo}^{S_2} r_{g_2}(b),$$

then the global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems. In $GEG_s(S_{\{G_1,G_2\}})$, we have:



4.2 Forcing Sharing Operations in Global Transactions

The extra operation method can also be utilized to enforce the fully sharing property on all global transactions, requiring only the insertion of retrieval operations. sharing serializability is therefore simpler and more efficient than chain conflicting serializability. Though the application of the fully sharing property to global transactions may sometimes burden them with long appendices, these will always be finite, since the data items in a local database are finite. Nevertheless, more elegant approaches need to be investigated. In the next section, we will show that such exponentially increasing appendices can be reduced automatically.

5 Hybrid Serializability

We will now discuss hybrid serializability which exhibits the characteristics both of chain conflicting and sharing serializability. The application of the hybrid property to global transactions offers a unique optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing restrictions on or requiring any information from that local site.

5.1 Hybrid Serializability

The definitions of hybrid transactions and hybrid serializable schedules clarify the manner in which they effectively combine the best features of chain conflicting serializability and sharing serializability.

Definition 8 (Hybrid transactions) *A set T of local transactions is hybrid if there is a total order $T_{i_1}, T_{i_2}, \cdots, T_{i_n}$ on T such that $T_{i_1} \Diamond T_{i_2} \Diamond \cdots \Diamond T_{i_n}$ where $\Diamond \in \{\overset{c}{\sim}, \subseteq, \supseteq\}$[5]. A set $\mathcal{G}$ of global transactions is hybrid if there is a total order O on $\mathcal{G}$ such that for all k, $1 \leq k \leq m$, $\mathcal{G}_k$ is hybrid in an order consistent with O.*

Definition 9 (Hybrid equivalence) *Two global subschedules $S_\mathcal{G}$ and $S'_\mathcal{G}$ of global schedule S and S' are said to be hybrid equivalent, denoted $S_\mathcal{G} \equiv_h S'_\mathcal{G}$, if they have the same operations of $\mathcal{G}$ where $\mathcal{G}$ is hybrid in a total order O on $\mathcal{G}$ and for any G_i preceding G_j in O, the following conditions are satisfied for all integer k $(1 \leq k \leq m)$:*

- *if $G_{ik} \overset{c}{\sim} G_{jk}$, then for all conflicting operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_\mathcal{G}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_\mathcal{G}} o_{jk}$; or*
- *if $G_{ik} \subseteq G_{jk}$ (or $G_{ik} \supseteq G_{jk}$), then for all sharing operations $o_{ik} \in OP_{G_{ik}}, o_{jk} \in OP_{G_{jk}}$, $o_{ik} \prec_{eo}^{S_\mathcal{G}} o_{jk}$ and $o_{ik} \prec_{eo}^{S'_\mathcal{G}} o_{jk}$.*

Definition 10 (Hybrid serializability) *A global subschedule is hybrid serializable if and only if it is hybrid equivalent to a serially global subschedule.*

Note that hybrid serializability is stronger than serializability; in other words, hybrid serializability implies serializability.

We will now illustrate the application of hybrid serializability in the MDBS environment.

[5]We consider that $\overset{c}{\sim}$ has higher priority to be chosen than $\subseteq$ (or $\supseteq$); in other words, if two transactions T_i and T_j have both $T_i \overset{c}{\sim} T_j$ and $T_i \subseteq$ (or $\supseteq$) T_j properties, then $T_i \overset{c}{\sim} T_j$ will be chosen in the hybrid ordering instead of $T_i \subseteq$ (or $\supseteq$) T_j. Both $\subseteq$ and $\supseteq$ have the same priority.

Theorem 6 *Let S be a global schedule and $\mathcal{G}$ be the set of committed global transactions in S. If $S_{\mathcal{G}}$ is hybrid serializable, then the local serializability of S_k (for k=1,...,m) implies the global serializability of S.*

The proof of this theorem is comparable to that of Theorem 2 and 4.

Following Theorem 6, global serializability can be achieved at the global level by controlling the execution order of global transactions for a special class of global transactions which is hybrid. In addition, only hybrid operations need to be ordered.

A global transaction execution graph of $S_{\mathcal{G}}$ in O for hybrid serializability, denoted $GEG_h(S_{\mathcal{G}}^O)$, can be defined by combining the conditions set forth in Definition 3 and Definition 7; a similar global execution theorem can also be provided. Rather than reiterating these formulations, we provide the following illustrative example:

Example 5 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b, c are in D_2. The following global transactions are submitted:*

$G_1 : w_{g_1}(a)r_{g_1}(b)$,

$G_2 : r_{g_2}(a)w_{g_2}(c)r_{g_2}(b)$,

$G_3 : r_{g_3}(a)r_{g_3}(c)r_{g_3}(b)$,

$G_4 : w_{g_4}(a)r_{g_4}(c)$,

which is hybrid in the order $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4$ where at local site LS_1, $G_{11} \overset{c}{\sim} G_{21} \subseteq G_{31} \overset{c}{\sim} G_{41}$ and at local site LS_2, $G_{12} \subseteq G_{22} \overset{c}{\sim} G_{32} \supseteq G_{42}$. Let $L_{2,1}$ be a local transaction submitted at local site LS_2:

$L_{2,1}$: $w_{L_{2,1}}(b)r_{L_{2,1}}(c)$

Let $S = \{S_1, S_2\}$ be the global schedule:

$S_1 : w_{g_1}(a)r_{g_2}(a)r_{g_3}(a)w_{g_4}(a)$

$S_2 : w_{L_{2,1}}(b)r_{g_1}(b)r_{L_{2,1}}(c)w_{g_2}(c)r_{g_3}(c)r_{g_3}(b)r_{g_4}(c)r_{g_2}(b)$.

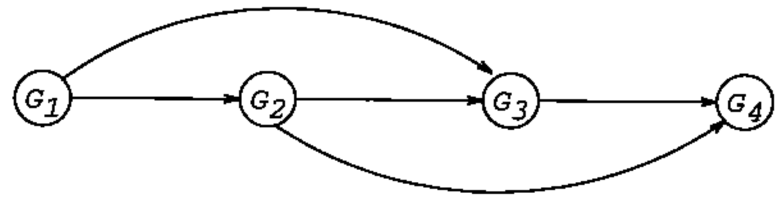
The global subschedule $S_{\mathcal{G}}$ is hybrid serializable in the order $G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_4$, and S is serializable. Note that, if the execution order of key operations which determine the hybrid relationships among global transactions are maintained:

$$w_{g_1}(a) \prec_{eo}^{S_1} r_{g_2}(a) \prec_{eo}^{S_1} r_{g_3}(a) \prec_{eo}^{S_1} w_{g_4}(a)$$

$$r_{g_1}(b) \prec_{eo}^{S_2} r_{g_2}(b)$$

$$w_{g_2}(c) \prec_{eo}^{S_2} r_{g_3}(c) \prec_{eo}^{S_2} r_{g_4}(c),$$

then global serializability is always maintained, even if local sites produce different local serializable schedules from the above. Local indirect conflicts will no longer create problems. In $GEG_h(S_{\mathcal{G}}^O)$, we have:



□

5.2 Optimality

The application of the hybrid property to global transaction scheduling provides a unique optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site. This is formally proven in the following theorem:

Theorem 7 (Optimality theorem) *For the systems using the events described in this paper, the hybrid property of global transactions is a unique optimal condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing any restrictions on or requiring any information from local sites.*

Proof: We first prove that the hybrid property is optimal. The proof proceeds by contradiction. Suppose the hybrid property of global transactions is not optimal. There then exists a property P of global transactions that is strictly weaker than the hybrid property, and the serialization order of global subtransactions at a local site is determined at global level. A counter case shows, however, that such a property does not exist.

Suppose, at a local site LS_k, there are only two global subtransactions G_{1k}, G_{2k} with $G_{1k} \not\stackrel{\circ}{\succ} G_{2k}$, $G_{1k} \not\subseteq G_{2k}$ and $G_{1k} \not\supseteq G_{2k}$. Then there are $op_1(x) \in OP_{G_{1k}}$ and $op_2(y) \in OP_{G_{2k}}$ such that $op_1(x)$, $op_2(y)$ do not conflict with each other and x, y refer to different data items. We construct a local transaction $L_1 : w(x)w(y)$. If the local scheduler produces the following schedule:

$$S_k \; : \; w(x)G_{1k}G_{2k}w(y),$$

then the serialization order of S_k is $G_{2k} \to L_1 \to G_{1k}$. On the other hand, if the local scheduler produces the following schedule:

$$S_k \; : \; G_{1k}w(x)w(y)G_{2k},$$

then the serialization order of S_k is $G_{1k} \to L_1 \to G_{2k}$. Consequently, the serialization order of S_k responds dynamically to the interactions of local transaction and global subtransactions, even though the execution order of global subtransactions remains consistent in both cases.

The generality of the above counter example of global subtransactions also implies that for any set of global transactions which is not hybrid, the serialization order of its subtransactions at a local site may not be determined at the global level. Hence, the hybrid property is a unique optimal property. □

Therefore, no other property of global transactions can be strictly weaker than the hybrid property to be applied as a sufficient condition for the GDBS to indirectly determine the serialization order of global subtransactions at a local site without imposing any restrictions on or requiring any information from local sites.

5.3 Forcing the Hybrid Property in Global Transactions

As pointed out earlier, the chain conflicting and fully sharing properties present the drawbacks of appending unnecessary updating operations or exponentially increasing appendices of extra retrieval operations. By combining the best features of these two properties, the hybrid property not only presents an optimal formulation but also offers a novel approach to compensating for the weakness of both previous methods. This is illustrated as follows:

Suppose we enforce the hybrid property on general global transactions by a particular order[6]. We append extra retrieval operations only if no hybrid order can be found between two global subtransactions. These appendices may render a subtransaction unwieldy, but they also increase the likelihood that it will conflict with or to be fully sharing with ($\supseteq$) the following subtransaction. Therefore, no extra operations may need be appended to the following subtransaction. The problem of exponentially increasing appendices is thus automatically avoided. The following example details these concepts.

Example 6 *Consider an MDBS consisting of two LDBSs on D_1 and D_2, where data item a is in D_1, and b, c, d are in D_2. The following non-global hybrid global transactions are submitted to the GDBS in the order G_1, G_2, G_3, G_4, G_5:*

$G_1 : w_{g_1}(a)r_{g_1}(b)$ $\quad$ $G_2 : r_{g_2}(a)r_{g_2}(c)$ $\quad$ $G_3 : r_{g_3}(a)r_{g_3}(d)$ $\quad$ $G_4 : w_{g_4}(a)r_{g_4}(b)$
$G_5 : r_{g_5}(a)w_{g_5}(b)$,

after appending extra retrieval operations in first-come-first-serve order, we get:

$$\left.\begin{array}{l} G_1 : w_{g_1}(a)r_{g_1}(b), \\ G_2 : r_{g_2}(a)r_{g_2}(c)\underbrace{r_{g_2}(b)}_{appended}, \\ G_3 : r_{g_3}(a)r_{g_3}(d)\underbrace{r_{g_3}(b)r_{g_3}(c)}_{appended}, \end{array}\right\} \textit{increasing appendices}$$

$$\begin{array}{l} \left.G_4 : w_{g_4}(a)r_{g_4}(b),\qquad\qquad\right\} \textit{reducing appendices} \\ G_5 : r_{g_5}(a)w_{g_5}(b), \end{array}$$

[6] This may be either first-come-first-serve, which enforces a hybrid order identical to the submitting order, or best-fit, which groups the global transactions and determines the most efficient hybrid order.

which is hybrid in the order $G_1 \to G_2 \to G_3 \to G_4 \to G_5$ *where at site* LS_1, $G_{11} \stackrel{c}{\sim} G_{21} \subseteq (or \supseteq) G_{31} \stackrel{c}{\sim} G_{41} \stackrel{c}{\sim} G_{51}$ *and at site* LS_2, $G_{12} \subseteq G_{22} \subseteq G_{32} \supseteq G_{42} \stackrel{c}{\sim} G_{52}$. □

Typically, the phases involving increasing and reducing appendices alternate, thus avoiding the spectre of exponentially increasing appendices. Furthermore, no extra updating operation needs to be appended to global transactions.

The extra operation method, in this paper, is only a theoretical tool to show that the hybrid property can be generalized to all global transactions. Due to space limitations, a detailed analysis of forcing the hybrid property and approaches to maintaining the hybrid serializability of global subschedules will appear elsewhere.

6 Relationship to Other Work

Many approaches have been proposed to solving the global concurrency control problem in MDBSs. Among them, two-level serializability and quasi-serializability characterize two correctness criteria for global schedules which maintain global consistency without imposing any restrictions on local sites[7]. In this section, we compare the present work with these two correctness criteria.

Let $\mathcal{H}$ denote the set of all possible global schedules; TSR denotes the set of two-level serializable global schedules; QSR denotes the set of quasi-serializable global schedules; SR denotes the set of serializable global schedules; CSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are chain conflicting serializable; SSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are sharing serializable; HSR denotes the set of serializable global schedules in which the global subschedules of the global schedules are hybrid serializable.

As stated in [MRKS91] and [DE89], TSR is a superset of QSR, and QSR is a superset of SR. As pointed out earlier in this paper, HSR is a subset of SR, and a superset of both CSR and SSR. There is no inclusion relationship between CSR and SSR. Note that the set of global schedules generated by the Optimistic Ticket Method (OTM) [GRS91] is a subset of CSR. Figure 2 depicts the relationships among these different types of global schedules.

If the set of all global transactions submitted at the global level is chain conflicting, the problem of global transaction scheduling is further reduced to maintaining serializability of global transactions in a certain order. This is a sufficient condition for two-level serializability, which then maintain global serializability. Thus, enforcing hybrid property on global transactions simplifies the global concurrency control problem and global serializability is still retained.

[7] Definitions of two-level serializability and quasi-serializability may be found in [MRKS91] and [DE89].

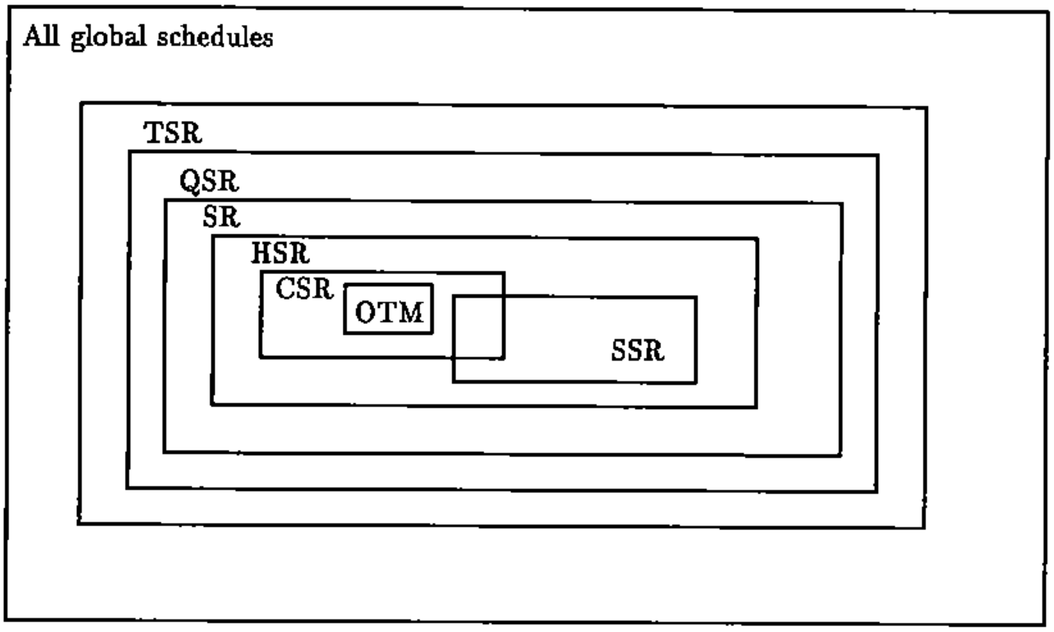


Figure 2: The relationships among TSR, QSR, SR, HSR, CSR, SSR, OTM

7 Conclusion

There has been no theoretical study of global transaction scheduling to maintain global serializability in MDBS environments. Existing theories for global concurrency control in MDBSs either relax the serializability theory or impose restrictions on local concurrency control protocols. In this paper, we have proposed three global transaction scheduling criteria to maintain global serializability without imposing any additional restriction on LDBSs except local serializability. These three criteria are chain conflicting serializability, sharing serializability, and hybrid serializability.

We have therefore:

- Formally proposed and proved a theory of global transaction scheduling for maintaining global serializability in multidatabase systems without placing any additional restrictions at local sites except local serializability;
- Indicated the upper limit on global serializability while maintaining local autonomy.

As an outgrowth of these criteria, we have shown that global serializability can be ensured at the global level by utilizing the intrinsic characteristics of global transactions and controlling their execution. We have also shown that global concurrency may be limited if local autonomy is a major factor to be considered in MDBSs.

Hybrid serializability formulates the maximal set of global schedules to be determined in MDBSs without violating local autonomy, which clarifies our view on how much the global concurrency controller can achieve if local autonomy needs to be maintained. Further work on the algorithmic aspect of the theory needs to be provided and also more work needs to be done on failure prone MDBS environment.

Acknowledgements

The authors has benefited greatly from the discussions with Xiangning Liu, Jiansan Chen, and Jim Mullen. We specially thank Xiangning Liu for spending time to check the paper and pointing out some minor errors in an earlier version of this paper.

References

[AGMS87] R. Alonso, H. Garcia-Molina, and K. Salem. Concurrency control and recovery for global procedures in federated database systems. In *IEEE Data Engineering Bulletin*, pages 5–11, September 1987.

[BGRS91] Y. Breitbart, D. Georgakopolous, M. Rusinkiewicz, and A. Silberschatz. On Rigorous Transaction Scheduling. *IEEE Transactions on Software Engineering*, 17(9), 1991.

[BHG87] P. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Databases Systems.* Addison-Wesley Publishing Co., 1987.

[BS88] Y. Breitbart and A. Silberschatz. Multidatabase update issues. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 135–142, June 1988.

[BS92] Y. Breitbart and A. Silberschatz. Strong recoverability in multidatabase systems. In *Proc. of the 2nd International Workshop on Research Issues on Data Engineering : Transaction and Query Processing*, pages 170–175, Tempe, AR, 1992.

[DE89] W. Du and A. Elmagarmid. Quasi Serializability: a Correctness Criterion for Global Concurrency Control in InterBase. In *Proceedings of the International Conference on Very Large Data Bases*, pages 347–355, Amsterdam, The Netherlands, August 1989.

[DELO89] W. Du, A. Elmagarmid, Y. Leu, and S. Ostermann. Effects of Autonomy on Global Concurrency Control in Heterogeneous Distributed Database Systems. In

Proceedings of the Second International Conference on Data and Knowledge Systems for Manufacturing and Engineering, pages 113–120, Gaithersburg, Maryland, October 1989.

[ED90] A. Elmagarmid and W. Du. A paradigm for concurrency control in heterogeneous distributed database systems. In *Proceedings of the Sixth International Conference on Data Engineering*, Los Angeles, California, February 1990.

[GMK88] H. Garcia-Molina and B. Kogan. Node Autonomy in Distributed Systems. In *Proceedings of the First International Symposium on Databases for Parallel and Distributed Systems*, pages 158–166, 1988.

[GPZ86] V. Gligor and R. Popescu-Zeletin. Transaction management in distributed heterogeneous database management systems. *Information Systems*, 11(4):287–297, 1986.

[GRS91] D. Georgakopoulos, M. Rusinkiewicz, and A. Sheth. On serializability of multidatabase transactions through forced local conflicts. In *Proceedings of the 7th Intl. Conf. on Data Engineering*, pages 314–323, Kobe, Japan, April 1991.

[Had88] V. Hadzilacos. A theory of reliability in database systems. *Journal of the Association for Computing Machinery*, 35(1):121–145, January 1988.

[MRB+92] S. Mehrotra, R. Rastogi, Y. Breitbart, H. F. Korth, and A. Silberschatz. The concurrency control problem in multidatabases: Characteristics and solutions. Technical Report TR-91-37, University of Texas at Austin Department of Computer Science, 1992.

[MRKS91] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. Non-serializable executions in heterogenous distributed database systems. In *Proceedings of 1st International Conference on Parallel and Distributed Information Systems*, pages 245–252, December 1991.

[Pu88] C. Pu. Superdatabases for composition of heterogeneous databases. In *Proceedings of the International Conference on Data Engineering*, pages 548–555, February 1988.

[Vei90] J. Veijalainen. *Transaction Concepts in Autonomous Database Environments.* R. Oldenbourg Verlag, Germany, 1990.

Appendix A

The proof of Theorem 1:

(*if*) Assume that there exists a total order O on global transactions in S, and for every local site $LS_k (1 \leq k \leq m)$, the serialization order of global subtransactions in S_k is consistent with O. We construct the serialization graph SG for S, denoted $SG(S)$, as a directed graph whose nodes are the transactions in S and whose edges are all $T_i \rightarrow T_j (i \neq j)$ on both global and local transactions such that one of T_i's operations precedes and conflicts with one of T_j's operations in S. We need to prove that $SG(S)$ is acyclic.

Suppose there is a cycle in $SG(S)$. Without loss of generality, let the cycle be $T_1 \rightarrow T_2 \rightarrow \cdots \rightarrow T_k \rightarrow T_1 (k > 1)$. These edges imply that in S, T_1 appears before T_2, which appears before T_3, which appears $\cdots$ before T_k, which appears before T_1. Since each local subschedule of S is serializable and there is no conflict between local transactions at one site and local transactions (or global subtransactions) at another site, there must be a set of global transactions $\{G_{i_1}, G_{i_2}, \cdots, G_{i_r}\} \subseteq \{T_1, T_2, \cdots, T_k\}$ such that G_{i_1} precedes G_{i_2}, G_{i_2} precedes G_{i_3}, $\cdots$, G_{i_r} precedes G_{i_1}. There is, then no total order on global transactions such that G_{i_1} precedes G_{i_2}, G_{i_2} precedes G_{i_3}, $\cdots$, G_{i_r} precedes G_{i_1} at the same time. This is contradictory to our assumption. Hence, $SG(S)$ is acyclic. By the serialization theorem given in [BHG87], S is serializable.

(*only if*) Assume that S is serializable in a total order O. Then, for each local site LS_k $(1 \leq k \leq m)$, the serialization order of S_k is consistent with O. Let O' be O restricted to the global transactions in S. Consequently, the serialization order of global subtransactions at each local site LS_k $(1 \leq k \leq m)$ is consistent with O'. Hence, we prove the theorem. □